ИНФРАСТРУКТУРА ТК

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
"ИНФРАСТРУКТУРА ТК"

Платформа для автоматизации технологических процессов и управления производством WISECON

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРОГРАММЫ WiseSafetyPro

2026

Авторские права и товарные знаки:

Заявление:

Содержание данного документа было тщательно проверено на соответствие аппаратному и программному обеспечению, упомянутому в документе. Мы не можем полностью гарантировать точность содержания и исключить возможность ошибок и упущений. Однако данные в документе будут регулярно пересматриваться, техническая информация будет обновляться соответствующим образом, и все необходимые исправления будут включены в последующие версии.

Данный документ был подготовлен и распространяется вместе с соответствующей системой. Функции или возможности, описанные в этом документе, могут не полностью соответствовать поставляемой системе.

Пожалуйста, обратите внимание, что содержание этого документа может быть изменено без предварительного уведомления. Мы ценим ваше понимание.

Мы приветствуем ваши предложения по улучшению.

Назначение настоящего руководства:

Настоящее руководство описывает функции программы **WiseSafetyPro** и предназначено для того, чтобы помочь пользователям правильно использовать программное обеспечение.

Содержание

1. Введение в WiseSafetyPro

1.1. Назначение программного обеспечения

Программное обеспечение **WiseSafetyPro** (ранее SafetyPro) является интегрированной средой разработки (IDE), предназначенной для программирования, конфигурирования и управления контроллерами системы **WISECON**.

Программа предоставляет инженерам полный набор инструментов для создания логики управления системами противоаварийной защиты (ПАЗ) и другими критически важными приложениями. WiseSafetyPro позволяет выполнять аппаратную конфигурацию, разработку алгоритмов, диагностику системы и онлайн-мониторинг.

1.2. Соответствие стандарту IEC 61131-3

WiseSafetyPro полностью соответствует международному стандарту **IEC 61131-3**, который определяет архитектуру и языки программирования для программируемых логических контроллеров (ПЛК). Это обеспечивает стандартизированный подход к разработке и поддержку следующих языков:

- **LD** (Ladder Diagram) – релейно-контактные схемы;
- **FBD** (Function Block Diagram) – диаграммы функциональных блоков;
- **ST** (Structured Text) – структурированный текст;
- **SFC** (Sequential Function Chart) – последовательностные функциональные схемы.

Программное обеспечение WiseSafetyPro прошло оценку соответствия и сертифицировано как инструментальное средство класса **T3** согласно требованиям стандарта **IEC 61508-3**.

При использовании WiseSafetyPro инженеру не требуется вручную учитывать особенности троированной архитектуры и резервирования модулей контроллера — платформа абстрагирует эти детали, позволяя сосредоточиться на разработке логики безопасности.

1.3. Системные требования (аппаратные и программные)

Для корректной работы программного обеспечения WiseSafetyPro рабочая станция должна соответствовать следующим требованиям:

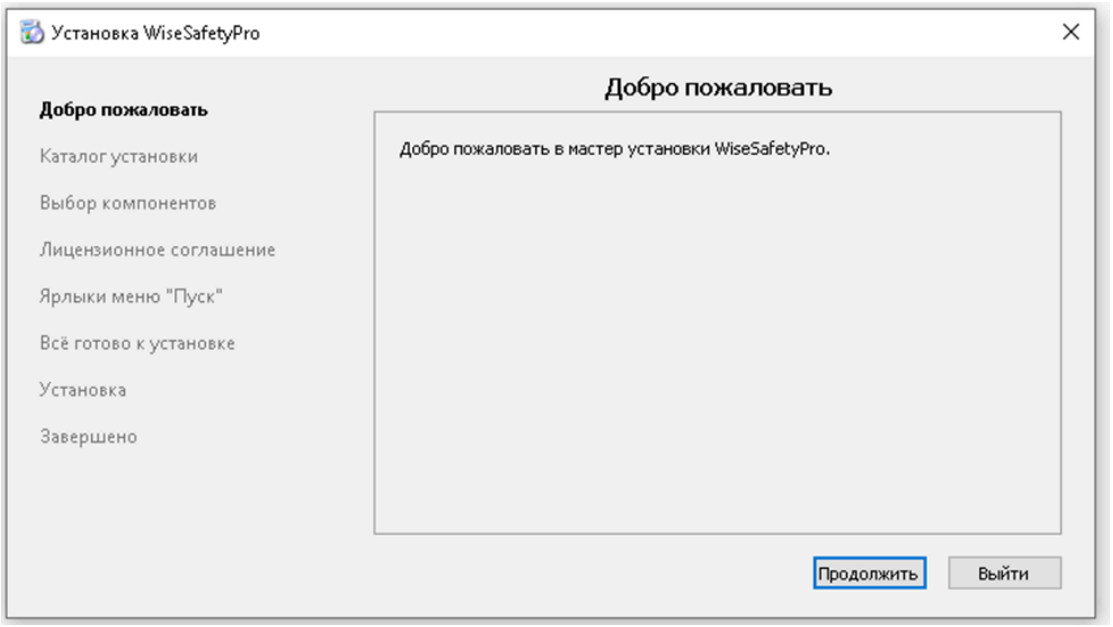
Компонент	**Минимальные требования**	**Рекомендуемые требования**
Операционная система	Windows 7/10/11, Astra Linux 1.7/1.8	Windows 10/11, Astra Linux 1.7/1.8
Оперативная память	4 ГБ	8 ГБ и более
Свободное место на диске	50 ГБ	200 ГБ и более

1.4. Установка и запуск программы

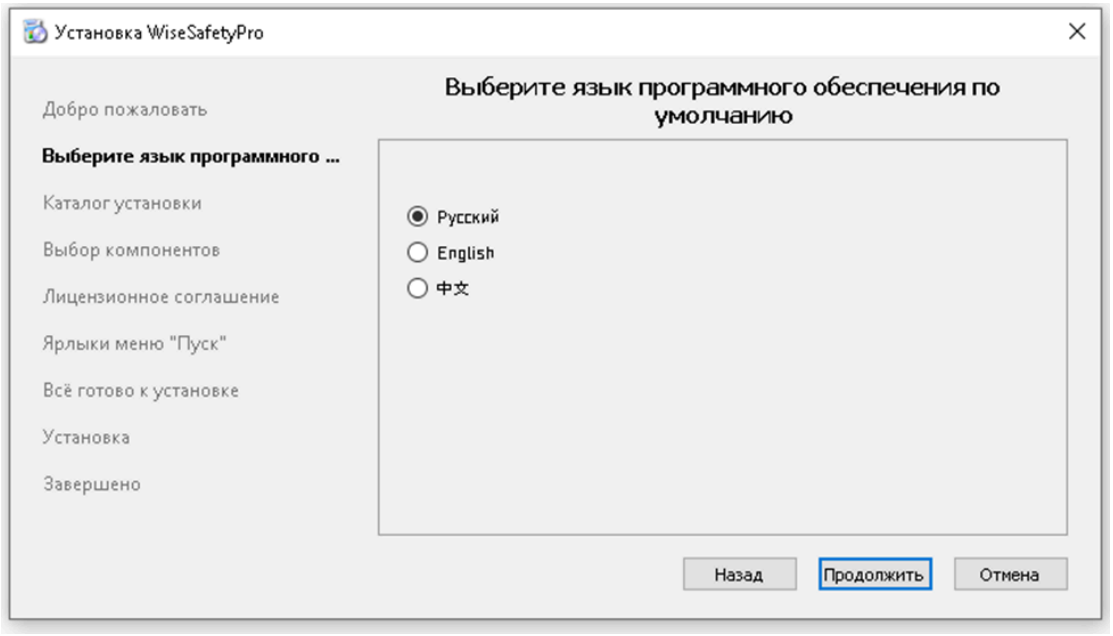
Установка

Процесс установки WiseSafetyPro выполняется с помощью стандартного мастера установки:

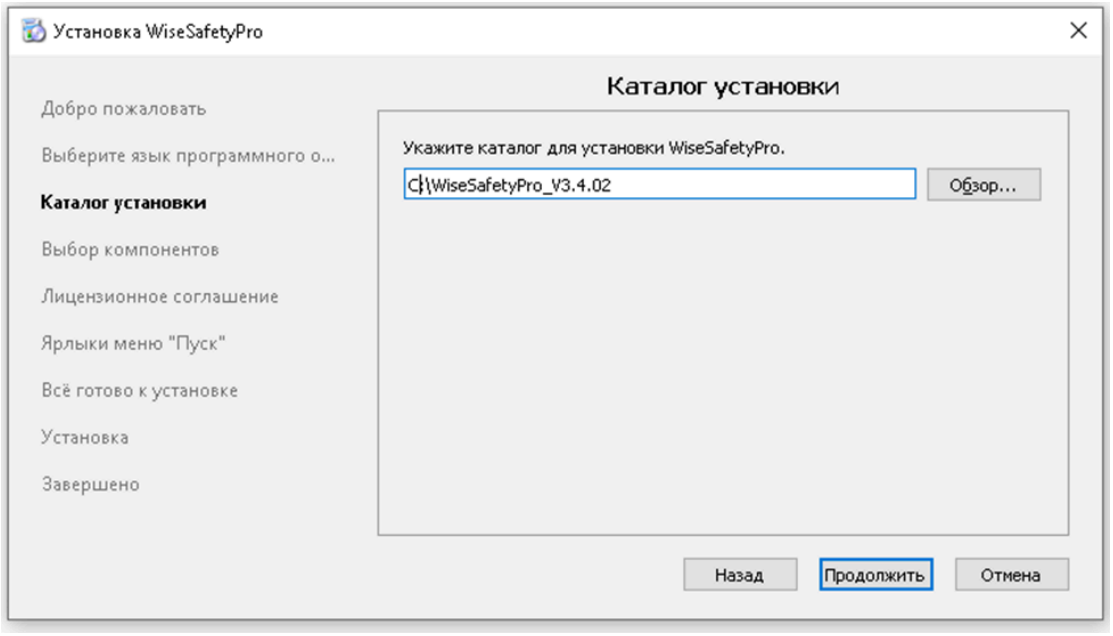
1. Запустите установочный файл.



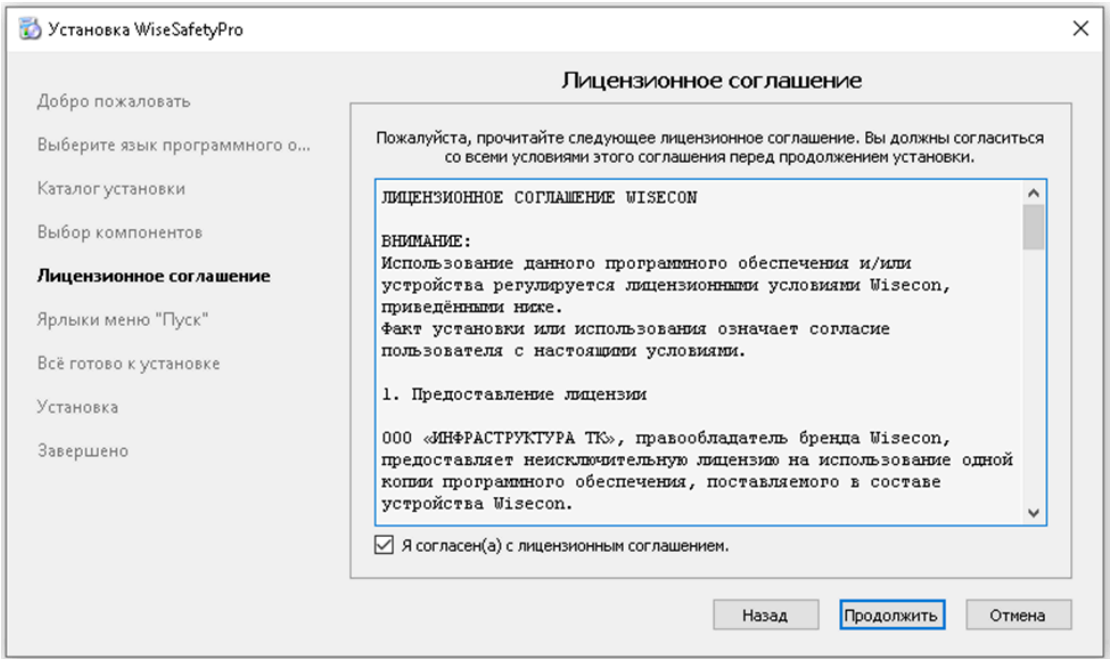
2. Выберите язык установки (например, русский).



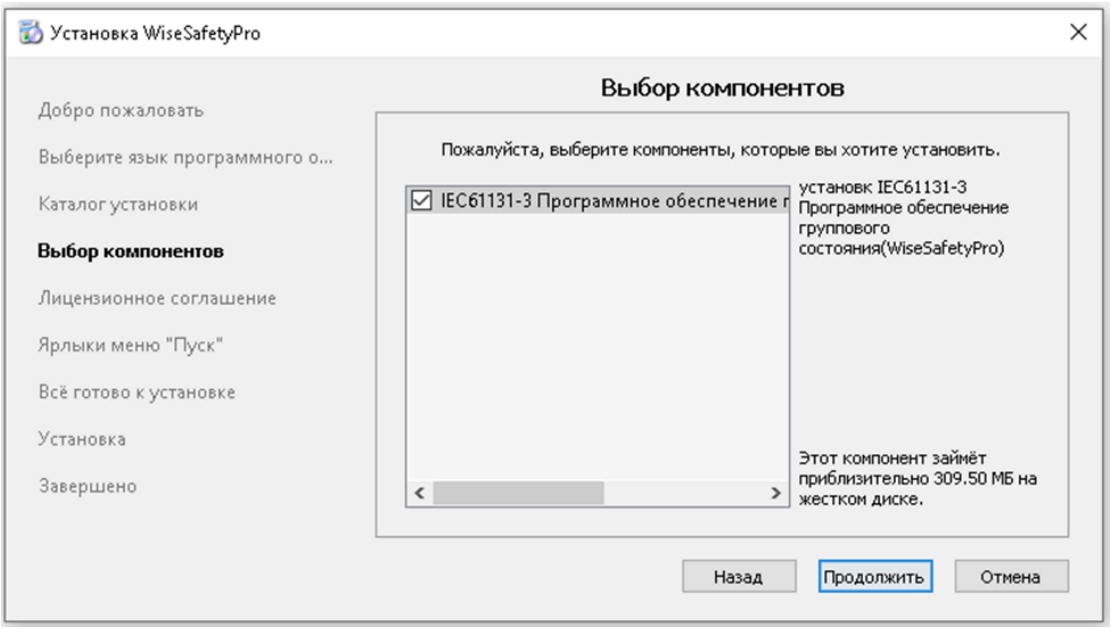
3. Выберите каталог для установки программы.



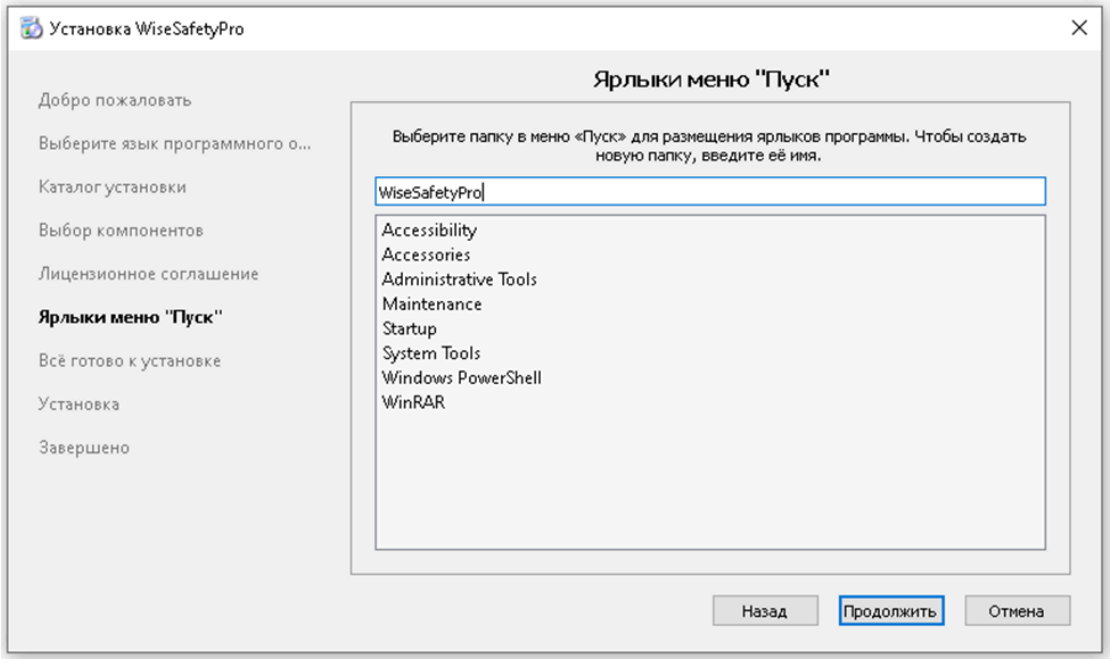
4. Примите условия лицензионного соглашения.



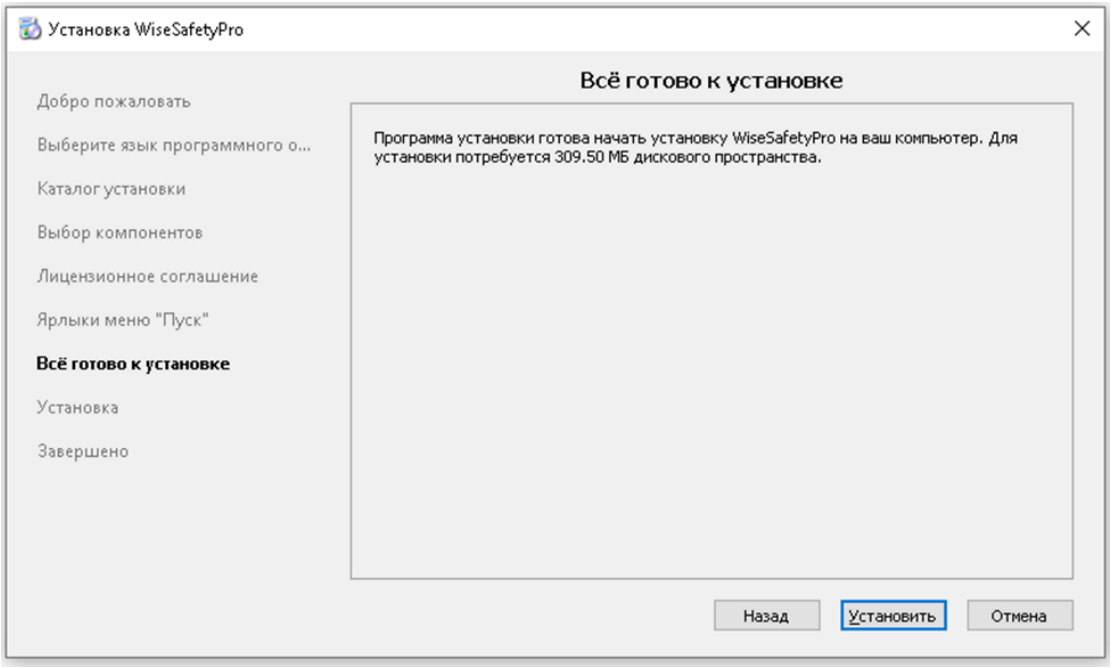
5. Выберите компоненты для установки (по умолчанию выбран основной компонент).



6. Создайте ярлык в меню «Пуск».



7. Следуйте дальнейшим инструкциям мастера для завершения установки.



Запуск

После завершения установки запустите программу одним из следующих способов:

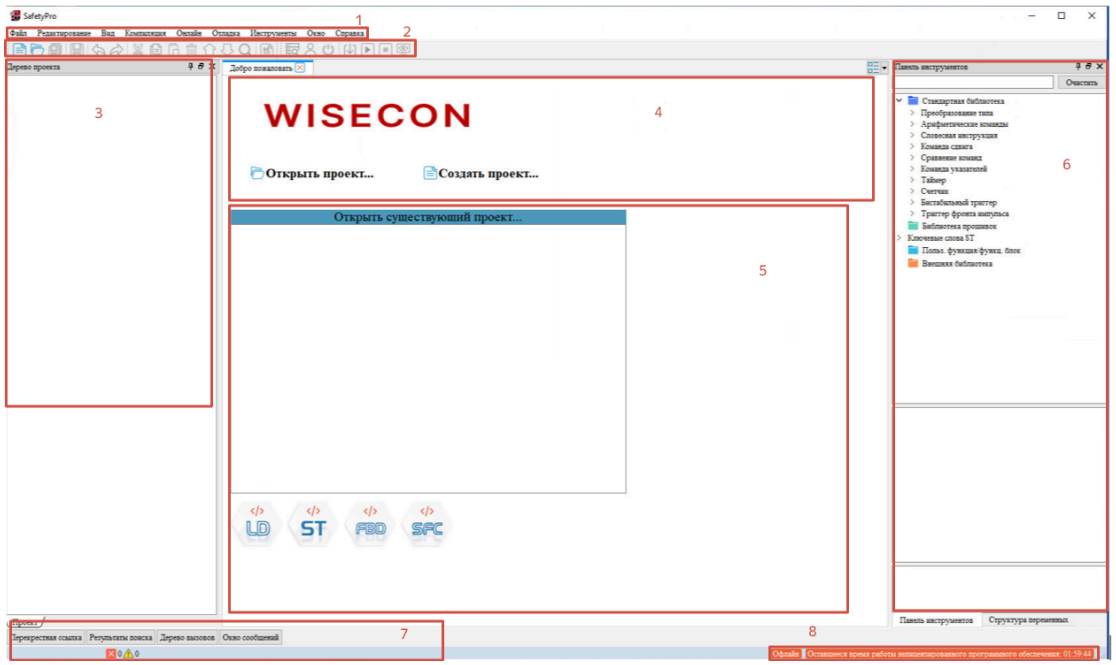
- Дважды щелкните по ярлыку **WiseSafetyPro** на рабочем столе.
- Выберите **WiseSafetyPro** в меню «Пуск».

При первом запуске откроется приветственная страница, на которой можно создать новый проект, открыть существующий или получить доступ к справочной документации.

2. Обзор среды программирования

Среда программирования **WiseSafetyPro** представляет собой многооконный интерфейс, разработанный для обеспечения удобства и эффективности при создании и отладке проектов систем безопасности.

2.1. Основной интерфейс: структура окон и панелей



Основной интерфейс программы включает в себя несколько ключевых элементов, которые можно настраивать (перемещать, скрывать и закреплять) в соответствии с предпочтениями пользователя.

Номер	Название	Описание
1	Строка меню	Расположена в верхней части окна и предоставляет доступ ко всем функциям программы, сгруппированным по категориям (Файл, Правка, Вид, Сборка, Онлайн и т.д.).
2	Панель инструментов	Находится под строкой меню и содержит иконки для быстрого вызова наиболее часто используемых команд (создание, сохранение, компиляция, подключение к устройству).
3	Панель управления проектом	Обычно расположена слева и представляет собой иерархическое дерево всех компонентов проекта.
4	Область переменных	Используется для настройки пользовательских переменных конфигурации.
5	Область программирования	Для программирования пользовательской логики конфигурации

	конфигурации	
6	Модуль алгоритмов	Предоставляет широкий набор алгоритмических блоков.
7	Окно сообщений	Расположено в нижней части интерфейса и отображает информацию о компиляции, ошибки, предупреждения и другие системные сообщения.
8	Строка состояния	Находится в самом низу окна и отображает информацию о текущем состоянии соединения с контроллером, режиме работы и статусе лицензии.

2.2. Панель управления проектом (Project Manager)

Панель управления проектом является основным навигационным элементом в WiseSafetyPro. Она отображает все ресурсы проекта в виде древовидной структуры, обеспечивая быстрый доступ к любому из них.

Основные узлы дерева проекта:

- **Устройство (Device):** Корневой узел, представляющий контроллер. Здесь настраиваются его сетевые параметры и общие свойства.
- **Библиотеки (Library Manager):** Управление подключенными библиотеками стандартных и пользовательских функций.
- **Типы данных (DataTypes):** Раздел для создания пользовательских типов данных (структур, массивов).
- **Функции (Functions):** Список созданных пользователем функций (FC).
- **Функциональные блоки (Function Blocks):** Список созданных пользователем функциональных блоков (FB).
- **Программы (Programs):** Содержит основные программные модули (POU), в которых реализуется логика управления.
- **Конфигурация (Config):**
    - **Конфигурация устройства (Device Configuration):** Графический интерфейс для настройки аппаратного состава контроллера (шасси, модули).

    - **Глобальные переменные (Global Variables):** Таблицы для объявления переменных, доступных во всем проекте.
- **Ресурсы (Resource):**
    - **Конфигурация задач (Task Config):** Настройка циклов выполнения программ (например, периодические задачи с разным временем цикла).

2.3. Панель инструментов (Toolbox)

Панель инструментов предоставляет доступ к элементам, которые используются для построения логических схем. Все элементы сгруппированы по библиотекам.

- **Standard.lib:** Стандартная библиотека, соответствующая IEC 61131-3. Включает в себя:
    - **Преобразования типов:** Блоки для конвертации данных (например, INT_TO_REAL).
    - **Математические инструкции:** Арифметические и тригонометрические функции (ADD, SIN, COS).
    - **Побитовые операции:** Блоки для логических операций (AND, OR, XOR).
    - **Сравнения:** Блоки для сравнения значений (GT, LT, EQ).
    - **Таймеры и счетчики:** Стандартные таймеры (TON, TOF) и счетчики (CTU, CTD).
- **Firmware.lib:** Библиотека, содержащая специализированные функциональные блоки, реализованные в прошивке контроллера. Включает в себя:
    - **Алгоритмы управления:** ПИД-регуляторы (PIDA, PIDI) и другие блоки для управления процессами.
    - **Логические инструкции:** Генераторы импульсов, триггеры и т.д.
    - **Системная диагностика:** Блоки для получения диагностической информации от системы.
- **Пользовательские библиотеки:** Библиотеки, созданные пользователем, которые содержат повторно используемые функции и функциональные блоки.

2.4. Область программирования конфигурации

Рабочая область является основным местом для разработки. В зависимости от выбранного элемента в дереве проекта, в ней открывается соответствующий редактор:

- **Редактор LD/FBD:** Графический редактор для языков релейно-контактных схем и функциональных блоков. Позволяет перетаскивать элементы из Toolbox и соединять их линиями связи.
- **Текстовый редактор ST:** Редактор для написания кода на языке структурированного текста с подсветкой синтаксиса и автодополнением.
- **Редактор аппаратной конфигурации:** Графический интерфейс для расстановки модулей на виртуальном шасси контроллера.
- **Редактор таблиц переменных:** Табличный интерфейс для объявления и настройки локальных и глобальных переменных.

2.5. Окно сообщений и строка состояния

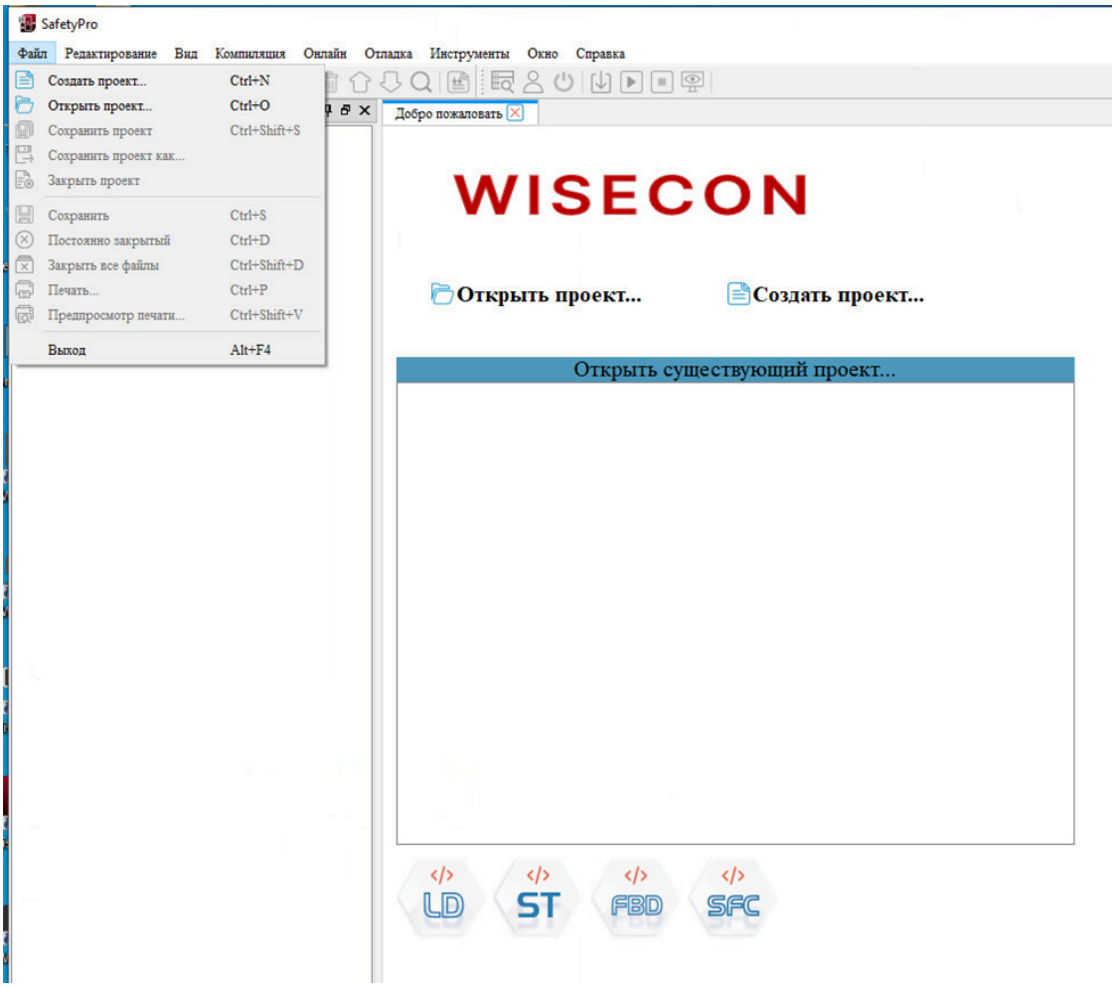
- **Окно сообщений:** Отображает результаты компиляции проекта, включая ошибки (Errors), предупреждения (Warnings) и информационные сообщения (Info). Двойной щелчок по сообщению об ошибке автоматически перемещает курсор к месту ее возникновения в коде или схеме.
- **Строка состояния:** Предоставляет оперативную информацию о:
    - **Статусе компиляции:** Указывает на наличие ошибок или предупреждений после последней сборки.
    - **Статусе устройства:** Показывает, подключена ли среда разработки к контроллеру и в каком он режиме (RUN, STOP).
    - **Статусе лицензии:** Информирует о состоянии лицензии на программное обеспечение.

3. Быстрый старт: Создание первого проекта

В этом разделе описаны основные шаги для создания простого проекта, который поможет новым пользователям быстро освоить базовые принципы работы в среде WiseSafetyPro.

3.1. Создание нового стандартного проекта

1. Запустите программу **WiseSafetyPro**.
2. Перейдите в меню **«Файл» (File)** и выберите пункт **«Создать проект» (New Project)**. Откроется мастер создания нового проекта.



3. В окне мастера выберите **«Стандартный проект» (Project Standard)**.
4. Выберите из списка **«Устройство» (Device)** целевую модель контроллера (например, **WT610A**).
5. Выберите язык программирования по умолчанию для первого программного модуля (например, **Ladder Diagram (LD)**).
6. Заполните поля:
    - **Имя проекта (ProjectName):** Введите имя для вашего проекта, например, Project1.
    - **Путь (Path):** Укажите каталог для сохранения проекта.
    - **Имя устройства (DeviceName):** Введите имя для вашего контроллера, например, Device1.

7. Нажмите **«OK»**. Среда разработки автоматически создаст структуру проекта, которую можно увидеть в **Панели управления проектом**.

3.2. Создание программного модуля (POU)

Программный модуль (POU — Program Organization Unit) является основным контейнером для вашей логики. По умолчанию один POU уже создан. Чтобы добавить новый:

1. В **Панели управления проектом** щелкните правой кнопкой мыши по узлу **«Программы» (Programs)**.
2. В контекстном меню выберите **«Добавить POU» (Add POU)**.
3. В открывшемся диалоговом окне:
    - **Имя (Name):** Введите имя для нового модуля, например, `MainLogic`.
    - **Тип (Type):** Оставьте **«Программа» (Program)**.
    - **Язык (Language):** Выберите язык программирования, например, **LD** или **FBD**.
4. Нажмите **«OK»**. Новый модуль появится в дереве проекта под узлом «Программы». Дважды щелкните по нему, чтобы открыть его в рабочей области.

3.3. Объявление переменных

Каждый программный модуль имеет свою область для объявления локальных переменных.

1. Откройте созданный вами POU (`MainLogic`).
2. Верхняя часть рабочей области — это редактор переменных. Он имеет два режима: табличный и текстовый.
3. **В табличном режиме:**
    - Щелкните правой кнопкой мыши и выберите **«Вставить переменную» (Insert Variable)**.
    - В новой строке заполните столбцы:
        - **Имя (Name):** Имя переменной, например, `StartButton`.
        - **Тип данных (DataType):** Тип переменной, например, **BOOL** для дискретного сигнала.

- **Атрибут (VarType):** Оставьте **VAR** для внутренней переменной.

4. Аналогичным образом создайте еще одну переменную MotorOutput с типом **BOOL**.

3.4. Базовая логика на языке LD или FBD

Теперь создадим простую логику: при нажатии кнопки StartButton включается MotorOutput.

На языке LD (релейно-контактные схемы):

1. Нижняя часть рабочей области — это редактор логики. Убедитесь, что у вас есть пустая сеть (Network).
2. На панели инструментов выберите инструмент **«Контакт» (Contact)** и щелкните на линии в сети, чтобы разместить его.
3. Над контактом появится поле для ввода переменной. Введите StartButton.
4. На панели инструментов выберите инструмент **«Катушка» (Coil)** и разместите ее в конце сети.
5. Над катушкой введите имя переменной MotorOutput.
6. Соедините контакт и катушку горизонтальной линией.

На языке FBD (диаграммы функциональных блоков):

1. В редакторе логики щелкните правой кнопкой мыши и выберите **«Вставить блок» (Insert Box)**.
2. В открывшемся окне найдите и выберите блок **MOVE** (пересылка значения).
3. К входу **EN** блока MOVE подключите переменную StartButton. Это можно сделать, щелкнув правой кнопкой мыши на входе и выбрав «Присвоить».
4. К входу **IN** блока MOVE присвойте константу TRUE.
5. К выходу блока **OUT** присвойте переменную MotorOutput.
6. Теперь, когда StartButton станет TRUE, блок MOVE будет активирован и присвоит TRUE переменной MotorOutput.

3.5. Конфигурация задач (Task Configuration)

Чтобы созданная программа выполнялась в контроллере, ее необходимо привязать к задаче.

1. В **Панели управления проектом** раскройте узел **«Ресурсы» (Resource)** и дважды щелкните по элементу **«Конфигурация задач» (Task Config)**.
2. Откроется список задач. По умолчанию существует одна циклическая задача (например, **MainTask**).
3. Выберите задачу и щелкните правой кнопкой мыши, чтобы вызвать контекстное меню, затем выберите **«Добавить вызов программы» (Add Program Call)**.
4. В открывшемся списке выберите вашу программу `MainLogic`.
5. В свойствах задачи (доступны по двойному щелчку) можно настроить ее тип (например, **Cyclic**) и интервал выполнения (например, `100ms`).

Теперь базовый проект готов. Его можно скомпилировать и загрузить в контроллер для проверки.

4. Конфигурация аппаратного обеспечения

Конфигурация аппаратного обеспечения (Hardware Configuration) — это процесс описания физической структуры системы управления в проекте WiseSafetyPro. Здесь определяется состав шасси (корзин), типы установленных модулей и их параметры. Корректная аппаратная конфигурация является обязательным условием для правильной работы контроллера и обмена данными с полевыми устройствами.

Доступ к редактору аппаратной конфигурации осуществляется через **Панель управления проектом** -> **Конфигурация**-> **Конфигурация устройства**.

4.1. Конфигурирование шасси (Rack Configuration)

Система WISECON, на которой работает WiseSafetyPro, может состоять из основного (локального) шасси и нескольких шасси расширения.

1. Добавление шасси расширения:

1. Откройте редактор **Конфигурация устройства**. По умолчанию в рабочей области отображается только основное (локальное) шасси.

2. Щелкните правой кнопкой мыши на пустом месте в рабочей области.
3. В контекстном меню выберите **«Добавить шасси» (Add Rack)**. На экране появится новое шасси расширения.
4. Можно добавить до 15 шасси расширения (с номерами от 0 до 14).

2. Настройка адреса шасси:

1. Щелкните правой кнопкой мыши по заголовку шасси расширения и выберите **«Редактировать шасси» (Edit Rack)**.
2. В открывшемся диалоговом окне укажите **номер шасси (Rack Number)**. Этот номер должен строго соответствовать положению DIP-переключателей на физическом шасси.

4.2. Добавление и настройка модулей (Module Configuration)

После того как структура шасси определена, необходимо заполнить ее модулями.

1. Добавление модуля:

1. В правой части окна находится **«Библиотека устройств» (Device Library)**, содержащая список всех доступных модулей (контроллеры, коммуникационные модули, модули ввода-вывода).
2. Выберите необходимый модуль из библиотеки и **перетащите** его мышью на нужный слот в графическом представлении шасси.
3. Альтернативный способ: щелкните правой кнопкой мыши по пустому слоту и выберите нужный модуль из контекстного меню.

2. Настройка модуля:

1. Выберите установленный модуль на шасси, чтобы его свойства отобразились в **Панели свойств** (обычно в нижней части экрана).
2. В панели свойств можно настроить основные параметры модуля, такие как:
    - **Режим резервирования (Redundancy mode):** Если модуль поддерживает резервирование, здесь можно включить или выключить этот режим.

- **Режим восстановления после сбоя (Fault recovery mode):** Устанавливает, будет ли модуль восстанавливать работу автоматически или потребует ручного сброса ошибки.
- **Подтверждение при включении (Power-on Confirm mode):** Определяет, как будет подтверждаться состояние модуля после включения питания.

3. Настройка каналов модуля:

1. В **Панели свойств** модуля перейдите на вкладку **«Конфигурация каналов» (Channel Config)**.
2. Отобразится таблица со всеми каналами данного модуля.
3. Для каждого канала можно настроить индивидуальные параметры:
   - **Имя переменной (Var Name):** Логическое имя канала, которое будет использоваться в программе.
   - **Тип сигнала (Signal Type):** Например, для аналогового входа это может быть "4-20mA" или "0-10V".
   - **Фильтрация (Filter time):** Время фильтрации для подавления шумов на аналоговых входах.
   - **Настройки SOE (SOE type):** Конфигурация регистрации событий для дискретных каналов (по переднему, заднему или обоим фронтам).
   - **Обработка при сбое (Fault Handling):** Поведение канала в случае обнаружения неисправности (например, удержание последнего значения).

4.3. Настройка коммуникационных параметров

Коммуникационные модули (например, **WT620A**) отвечают за связь контроллера с другими узлами сети. Их настройка является критически важным шагом.

1. Установите коммуникационный модуль в соответствующий слот на основном шасси.
2. Выберите модуль. В **Панели свойств** появятся его сетевые настройки.
3. Задайте следующие ключевые параметры:

- **Адрес домена (Domain Address):** Укажите номер домена (от 1 до 63). Этот номер должен соответствовать положению DIP-переключателей на физическом модуле.
- **Адрес станции (Site Address):** Укажите уникальный адрес станции в пределах домена (от 1 до 127), который также задается DIP-переключателями.
- **IP-адреса (NET-A-IP, NET-B-IP):** IP-адреса для каждого из двух сетевых портов (A и B). Они, как правило, формируются автоматически на основе адреса домена и станции, но могут быть скорректированы при необходимости.
- **Маски подсети (NET-A Subne, NET-B Subne):** Задайте маски подсети для каждого сетевого интерфейса.
- **Синхронизация времени (Clock synchronization):** Настройте параметры синхронизации времени по протоколам SNTP или NTP.
- **Настройки Modbus/OPC UA:** Если модуль поддерживает эти протоколы, здесь можно активировать и настроить соответствующие серверы.

5. Программирование логики

В этом разделе рассматриваются основные принципы создания логики управления в среде WiseSafetyPro с использованием поддерживаемых языков программирования стандарта IEC 61131-3.

5.1. Работа с переменными (создание, редактирование, типы данных)

Переменные (теги) являются основой любого проекта. Они используются для хранения данных, обмена информацией между программными блоками и связи с физическими входами/выходами контроллера.

Создание и редактирование переменных:
Переменные объявляются в верхней части редактора программного модуля (POU) в табличной или текстовой форме.

1. **Создание:**

- **Табличный режим:** Щелкните правой кнопкой мыши в области переменных и выберите **«Вставить переменную»**. В новой строке укажите **Имя**, **Тип данных** и другие атрибуты.
- **Текстовый режим:** Объявите переменную вручную между ключевыми словами `VAR` и `END_VAR`, например: `MyVariable : BOOL;`.

2. **Редактирование:** Для изменения параметров переменной дважды щелкните по соответствующей ячейке в таблице или отредактируйте текст в текстовом режиме.
3. **Привязка к аппаратному адресу:** Для связи переменной с физическим каналом модуля ввода-вывода в поле **«Адрес» (Address)** укажите прямой адрес, например, `%IX0.1.1.0` (дискретный вход) или `%IW0.1.2.0` (аналоговый вход).

Типы данных:
WiseSafetyPro поддерживает все стандартные типы данных IEC 61131-3, включая:

- **Дискретные:** `BOOL` (логическое значение, TRUE/FALSE)
- **Целочисленные:** `SINT`, `INT`, `DINT`, `LINT` (знаковые); `USINT`, `UINT`, `UDINT`, `ULINT` (беззнаковые)
- **С плавающей точкой:** `REAL`, `LREAL`
- **Временные:** `TIME`, `DATE`, `TOD` (время суток), `DT` (дата и время)
- **Строковые:** `STRING`

5.2. Программирование на языке Ladder Diagram (LD)

Язык LD (релейно-контактные схемы) является графическим языком, имитирующим электрические релейные цепи. Логика строится из сетей (networks), состоящих из контактов, катушек и функциональных блоков.

1. **Контакты:**
   - **Нормально открытый (`--| |--`):** Замыкается (пропускает сигнал), когда связанная с ним переменная `BOOL` имеет значение `TRUE`.

- **Нормально закрытый (`--|/|--`):** Замыкается, когда переменная BOOL имеет значение FALSE).
- Для вставки выберите иконку контакта на панели инструментов и щелкните на нужном месте в сети. Затем присвойте ему переменную.

2. **Катушки:**
   - **Обычная катушка (`--( )`):** Принимает значение TRUE, когда цепь перед ней замкнута.
   - **Установочная (Set) и сбросовая (Reset) катушки:** Устанавливают или сбрасывают BOOL переменную в TRUE или FALSE соответственно.
   - Вставляются аналогично контактам в конце логической цепи.
3. **Функциональные блоки:**
   - Для вставки таймеров, счетчиков, математических или других блоков перетащите их из **Панели инструментов (Toolbox)** в сеть.
   - Входы и выходы блока подключаются к переменным или другим элементам схемы.

- Логика выполняется слева направо и сверху вниз.
- Каждая сеть (network) представляет собой законченную логическую цепь.
- Для создания параллельных ветвей используйте инструменты для вставки ветвей выше или ниже текущей линии.

5.3. Программирование на языке Function Block Diagram (FBD)

FBD (диаграммы функциональных блоков) — это графический язык, где программа представлена в виде набора блоков, соединенных линиями связи.

1. **Вставка блока:**
   - Перетащите необходимый блок (например, ADD для сложения, TON для таймера) из **Панели инструментов (Toolbox)** в рабочую область.
   - Также можно щелкнуть правой кнопкой мыши и выбрать **«Вставить блок»**.
2. **Присвоение переменных:**

- К входам и выходам блока можно привязать переменные. Щелкните на поле рядом с входом/выходом и введите имя переменной.

3. **Соединение блоков:**
   - Для соединения выхода одного блока со входом другого щелкните мышью на выходе, а затем, удерживая левую кнопку, протяните линию до нужного входа. Программа автоматически создаст соединение.

В отличие от LD, порядок выполнения блоков в FBD не всегда очевиден. По умолчанию система определяет его автоматически на основе потока данных. Для принудительного задания последовательности можно использовать специальные элементы **«Порядок выполнения»**, которые позволяют явно указать, какой блок должен выполняться раньше.

5.4. Использование инструкций на языке Structured Text (ST)

Язык ST (структурированный текст) — это текстовый язык высокого уровня, похожий на Pascal. Он идеально подходит для реализации сложных математических алгоритмов, циклов и условных переходов.

Инструкции ST можно использовать внутри графических языков LD и FBD, вставив специальный **«Блок ST»**.

Основные конструкции:

- **Присваивание:** `Variable := Expression;`
  - *Пример:* `MotorSpeed := 1500.0;`

- **Условный оператор:**

```
IF Condition THEN
    Statements;
ELSIF Another_Condition THEN
```

```
    Other_Statements;
ELSE
    Else_Statements;
END_IF;
```

- **Циклы:**
  - **Цикл FOR:** FOR i := 1 TO 10 BY 1 DO ... END_FOR;
  - **Цикл WHILE:** WHILE Condition DO ... END_WHILE;

- **Вызов функциональных блоков:**

```
MyTimer(IN:= StartSignal, PT:= T#10s);
TimerOutput := MyTimer.Q;
```

Для вставки кода ST в графический редактор выберите на панели инструментов **«Вставить блок ST» (Insert ST Box)** и введите необходимый код в открывшемся окне.

6. Работа с библиотеками

Библиотеки в WiseSafetyPro — это наборы готовых к использованию программных компонентов (функций и функциональных блоков), которые значительно ускоряют и упрощают процесс разработки. Они позволяют повторно использовать проверенную логику и стандартизировать решения.

6.1. Обзор системных и пользовательских библиотек

Все доступные в проекте библиотеки отображаются на **Панели инструментов (Toolbox)**.

- **Системные библиотеки:**
    - **Standard.lib:** Стандартная библиотека, определенная стандартом IEC 61131-3. Она содержит базовый набор элементов: математические операции (ADD, MUL), логические блоки (AND, OR), компараторы (GT, LT, EQ), таймеры (TON, TOF) и счетчики (CTU, CTD). Эти блоки являются универсальными и доступны в любом проекте.
- **Пользовательские библиотеки:**
    - Это библиотеки, созданные пользователем или сторонними разработчиками. Они могут содержать специализированные алгоритмы, адаптированные под конкретные технологические процессы или оборудование. Пользовательские библиотеки можно добавлять в проект, экспортировать для использования в других проектах и распространять между инженерами.

6.2. Создание и экспорт/импорт пользовательских библиотек

WiseSafetyPro предоставляет инструменты для создания собственных библиотек, что позволяет инкапсулировать часто используемую логику в виде удобных для повторного использования блоков.

Создание пользовательской библиотеки:

1. Создайте новый проект, выбрав в мастере создания тип **«Библиотека» (Library Project)**.
2. В этом проекте вы не сможете конфигурировать аппаратное обеспечение или задачи. Вместо этого вы можете создавать новые **Функции (Functions)** и **Функциональные блоки (Function Blocks)**.
3. Разработайте необходимую логику внутри созданных POU, определив их входы (VAR_INPUT), выходы (VAR_OUTPUT) и внутренние переменные (VAR).
4. После завершения разработки сохраните проект.

Экспорт библиотеки:

1. Откройте проект библиотеки, который вы хотите экспортировать.

2. Перейдите в меню **«Файл» (File)** и выберите **«Экспорт библиотеки» (Export Library)**.
3. Укажите имя и место для сохранения файла библиотеки (с расширением **.sl**).

Импорт (добавление) библиотеки в проект:

1. Откройте стандартный проект, в который вы хотите добавить библиотеку.
2. В **Панели управления проектом** щелкните правой кнопкой мыши по узлу **«Библиотеки» (Library Manager)**.
3. В контекстном меню выберите **«Добавить библиотеку» (Add Library)**.
4. В диалоговом окне найдите и выберите ранее экспортированный файл библиотеки (.sl).
5. После добавления все функции и функциональные блоки из этой библиотеки появятся на **Панели инструментов (Toolbox)** в соответствующем разделе.

6.3. Использование функций и функциональных блоков из библиотек

Использование библиотечных элементов в программной логике является основной практикой программирования в WiseSafetyPro.

Процесс использования:

1. Откройте программный модуль (POU) в редакторе LD или FBD.
2. На **Панели инструментов (Toolbox)** найдите нужную библиотеку (например, Standard.lib) и раскройте ее.
3. Найдите необходимый функциональный блок (например, таймер TON или счетчик CTU).
4. **Перетащите** иконку блока мышью из панели инструментов в рабочую область (в сеть на LD или на поле в FBD).
5. Система автоматически предложит создать **экземпляр** блока. Экземпляр — это рабочая копия блока со своим уникальным именем и набором данных. Введите имя экземпляра (например, Timer_01) и нажмите «OK».
6. Подключите входы и выходы созданного экземпляра к переменным вашего проекта.

Различие между функцией (FC) и функциональным блоком (FB):

- **Функция (FC):** Не имеет "памяти". При каждом вызове она выполняет вычисления на основе текущих входных данных и возвращает один результат. Для нее не нужно создавать экземпляр.
- **Функциональный блок (FB):** Имеет "память" (внутренние переменные), состояние которой сохраняется между вызовами. Например, таймер помнит, сколько времени уже прошло. Поэтому для каждого использования FB необходимо создавать уникальный **экземпляр**.

7. Компиляция, загрузка и отладка

Этот раздел описывает процессы преобразования разработанного проекта в исполняемый код, загрузки его в контроллер и последующей отладки в режиме онлайн.

7.1. Компиляция проекта (инкрементальная и полная)

Компиляция — это процесс преобразования графических схем и текстового кода в машинный код, понятный контроллеру. WiseSafetyPro поддерживает два режима компиляции.

- **Инкрементальная компиляция (Increment Compile):**
  - **Назначение:** Перекомпилирует только те части проекта, которые были изменены с момента последней компиляции.
  - **Использование:** Нажмите кнопку **«Инкрементальная компиляция»** на панели инструментов или выберите соответствующий пункт в меню **«Сборка» (Build)**. Этот режим значительно быстрее и является предпочтительным в процессе разработки.
- **Полная компиляция (Compile All):**
  - **Назначение:** Полностью перекомпилирует весь проект с нуля.
  - **Использование:** Нажмите кнопку **«Полная компиляция»** на панели инструментов. Этот режим используется, когда необходимо гарантировать целостность всего проекта, например,

после значительных изменений в глобальных переменных, типах данных или перед финальной загрузкой.

В процессе компиляции все ошибки и предупреждения будут отображаться в **Окне сообщений**. Проект не может быть загружен в контроллер, если в нем есть ошибки компиляции.

7.2. Установка соединения с контроллером

Для загрузки проекта и онлайн-отладки необходимо установить соединение со средой исполнения (физическим контроллером или симулятором).

1. Убедитесь, что контроллер подключен к сети и доступен с вашего компьютера.
2. На панели инструментов нажмите кнопку **«Онлайн» (Online)** или выберите соответствующий пункт в меню.
3. Программа попытается установить соединение с контроллером, используя сетевые параметры, указанные в свойствах устройства в проекте.
4. В случае успеха в **Строке состояния** появится информация о подключении, а иконки онлайн-функций на панели инструментов станут активными.

7.3. Загрузка и выгрузка конфигурации

- **Загрузка (Download):**
    - **Назначение:** Передача скомпилированного проекта из WiseSafetyPro в память контроллера.
    - **Процесс:** После установки соединения нажмите кнопку **«Загрузить» (Download)** на панели инструментов. Система предложит остановить выполнение программы на контроллере (если она запущена) и выполнит загрузку. После завершения можно запустить программу.
    - **"Горячая" загрузка (Hot download):** WiseSafetyPro поддерживает загрузку изменений без полной остановки контроллера, что минимизирует прерывание технологического процесса.

- **Выгрузка (Upload):**
    - **Назначение:** Считывание конфигурации, находящейся в данный момент в контроллере, и ее загрузка в WiseSafetyPro.
    - **Использование:** Эта функция полезна для архивации текущей рабочей версии программы или для анализа конфигурации, если исходный файл проекта утерян.

7.4. Онлайн-мониторинг и отладка

WiseSafetyPro предоставляет мощные инструменты для наблюдения за работой программы в реальном времени и поиска ошибок в логике.

- **Назначение:** Отображение текущих значений переменных и состояния логических связей непосредственно на схемах LD и FBD или в коде ST.
- **Активация:** После установки соединения с контроллером нажмите кнопку **«Мониторинг/Отладка» (Monitor/Debug)**.
- **Отображение:**
    - Значения аналоговых и числовых переменных отображаются рядом с ними.
    - Логические цепи в LD и FBD подсвечиваются (обычно синим или зеленым цветом), когда по ним "протекает" сигнал (TRUE).
    - В таблице переменных отображаются их текущие значения.
- **Назначение:** Режим отладки позволяет приостанавливать выполнение программы в определенных местах и выполнять ее пошагово для детального анализа логики.
- **Точки останова (Breakpoints):**
    - **Установка/снятие:** Щелкните на левом поле редактора рядом с нужной строкой кода (в ST) или сетью (в LD/FBD), чтобы установить или снять точку останова.
    - **Срабатывание:** Когда выполнение программы доходит до точки останова, контроллер приостанавливает выполнение задачи, и программа "замирает" в этом месте.
- **Пошаговое выполнение:**
    - После остановки на точке останова становятся доступными кнопки пошагового выполнения:

        - **Шаг с обходом (Step Over):** Выполняет одну инструкцию или одну сеть, не заходя внутрь вызываемых функций или блоков.
        - **Шаг с заходом (Step In):** Если текущая инструкция — вызов функции или блока, переходит к первой строке кода внутри этого POU.
        - **Шаг с выходом (Step Out):** Выполняет оставшуюся часть текущего POU и останавливается на следующей инструкции после его вызова.
        - **Продолжить выполнение (Run):** Возобновляет нормальное выполнение программы до следующей точки останова или до остановки пользователем.
- **Назначение:** Позволяет вручную изменять значения переменных в контроллере во время его работы, игнорируя их физические входы или результаты вычислений.
- **Изменение значения:**
    - В режиме мониторинга дважды щелкните по значению переменной в таблице или на схеме.
    - Введите новое значение и подтвердите его. Это значение будет записано в контроллер однократно.
- **Форсирование (Force):**
    - Щелкните правой кнопкой мыши по переменной и выберите **«Форсировать» (Force)**.
    - Введите значение, которое будет принудительно поддерживаться в контроллере.
    - Форсированные переменные остаются неизменными до тех пор, пока форсирование не будет снято (**Unforce**).
    - *Внимание: Использование форсирования может быть опасным, так как оно отключает нормальную логику работы. Используйте эту функцию с особой осторожностью.*

8. Управление проектом и безопасность

WiseSafetyPro предоставляет многоуровневую систему защиты для обеспечения целостности проекта, защиты интеллектуальной собственности и разграничения доступа к функциям разработки и администрирования.

8.1. Управление пользователями и правами доступа

Система управления пользователями позволяет создавать различные учетные записи и группы с предопределенными наборами прав.

Создание и настройка пользователей:

1. В **Панели управления проектом** раскройте узел **«Инструменты» (Tool)** и дважды щелкните по **«Менеджер полномочий» (Authority Manager)**.
2. В открывшемся окне можно создавать новые группы пользователей (например, «Инженеры», «Операторы») и отдельных пользователей.
3. **Для каждой группы или пользователя можно детально настроить права доступа**, разрешая или запрещая выполнение определенных действий, таких как:
    - **Изменение аппаратной конфигурации.**
    - **Редактирование логики.**
    - **Загрузка/выгрузка проекта.**
    - **Онлайн-мониторинг и отладка.**
    - **Изменение параметров онлайн.**
    - **Управление пользователями.**

По умолчанию в системе существует пользователь **admin** с полными правами, для которого необходимо задать пароль при первой настройке безопасности.

Синхронизация с контроллером:
Настройки пользователей и прав, заданные в проекте, могут быть загружены в контроллер. После этого для подключения к контроллеру в онлайн-режиме (например, для мониторинга или загрузки) потребуется ввести соответствующий логин и пароль.

8.2. Защита паролем проекта и отдельных программных модулей (POU)

Помимо разграничения прав пользователей, WiseSafetyPro предлагает защиту паролем для проекта в целом и для отдельных его частей.

1. Пароль на проект:

- **Назначение:** Защищает весь проект от несанкционированного открытия.
- **Установка:** В настройках управления пользователями можно задать пароль для проекта. После установки при каждой попытке открыть файл проекта (.scpro) система будет запрашивать логин и пароль.

2. Шифрование программных модулей (POU Encryption):

- **Назначение:** Защищает интеллектуальную собственность, позволяя скрыть исходный код или логическую схему конкретной функции или функционального блока.
- **Процесс шифрования:**
    - В **Панели управления проектом** щелкните правой кнопкой мыши по POU, который необходимо защитить.
    - В контекстном меню выберите **«Зашифровать» (Encrypt)**.
    - Задайте пароль для данного POU.
- **Использование зашифрованного POU:**
    - Зашифрованный POU можно использовать в проекте (вставлять в схемы, вызывать), но его внутреннее содержимое будет недоступно для просмотра или редактирования без ввода правильного пароля.
    - Для снятия защиты необходимо выбрать POU, щелкнуть правой кнопкой мыши, выбрать **«Расшифровать» (Decrypt)** и ввести пароль.

8.3. Резервное копирование и восстановление конфигурации

Функция резервного копирования (бэкапа) позволяет создавать архивные копии конфигурации устройства для предотвращения потери данных и быстрого восстановления системы.

1. Создание резервной копии (Backup):

1. В **Панели управления проектом** щелкните правой кнопкой мыши по узлу устройства (Device).
2. В контекстном меню выберите **«Резервное копирование и восстановление» (Configuration Backup and Restore)**.
3. В открывшемся окне управления версиями нажмите кнопку **«Резервное копирование» (Backup)**.
4. Введите описание для создаваемой копии (например, «Версия после пусконаладки»).
5. Система создаст архивную копию текущей конфигурации проекта с меткой времени.
6. Также можно настроить **автоматическое резервное копирование** через заданные интервалы времени.

2. Восстановление из резервной копии (Restore):

1. Откройте окно **«Резервное копирование и восстановление»**.
2. В списке доступных резервных копий выберите ту версию, которую необходимо восстановить.
3. Нажмите кнопку **«Восстановить» (Restore)**.
4. Система предупредит, что текущая конфигурация будет заменена на выбранную из архива.
5. После подтверждения проект будет откачен к состоянию на момент создания резервной копии.

Эта функция является незаменимым инструментом для управления версиями проекта и восстановления системы после неудачных изменений или сбоев.

Приложение А. Основные функциональные блоки

В данном приложении представлен краткий справочник по наиболее часто используемым функциональным блокам из библиотек **Standard.lib**.

1. Стандартная библиотека (Standard.lib)

Таймеры являются основными элементами для организации задержек и временных интервалов в логике управления.

- **TON (Timer On-Delay) - Таймер с задержкой включения**
  - **Назначение:** Формирует выходной сигнал TRUE после того, как входной сигнал остается TRUE в течение заданного времени.
  - **Входы:**
    1. IN (BOOL): Вход запуска таймера.
    2. PT (TIME): Уставка времени задержки (например, T#5s для 5 секунд).
  - **Выходы:**
    1. Q (BOOL): Выход таймера.
    2. ET (TIME): Текущее отсчитанное время.
  - **Принцип работы:**
    1. Когда IN = FALSE, таймер сброшен: Q = FALSE и ET = T#0s.
    2. Когда IN переходит в TRUE, таймер начинает отсчет времени, увеличивая ET. Пока ET < PT, выход Q остается FALSE.
    3. Когда ET достигает значения PT, выход Q становится TRUE и остается в этом состоянии, пока IN = TRUE.
  - **Применение:** Задержка включения оборудования после получения команды, фильтрация кратковременных сигналов.
- **TOF (Timer Off-Delay) - Таймер с задержкой выключения**
  - **Назначение:** Удерживает выходной сигнал в состоянии TRUE в течение заданного времени после того, как входной сигнал стал FALSE.
  - **Входы:** IN (BOOL), PT (TIME).
  - **Выходы:** Q (BOOL), ET (TIME).

    - **Принцип работы:**
        1. Когда IN = TRUE, выход Q немедленно становится TRUE, а таймер ET сброшен (T#0s).
        2. Когда IN переходит в FALSE, таймер начинает отсчет времени. Выход Q остается TRUE, пока ET < PT.
        3. Когда ET достигает PT, выход Q становится FALSE.
    - **Применение:** Обеспечение выбега оборудования (например, работа вентилятора охлаждения после отключения основного устройства).
- **TP (Timer Pulse) - Формирователь импульса**
    - **Назначение:** Генерирует на выходе импульс заданной длительности по переднему фронту входного сигнала.
    - **Входы:** IN (BOOL), PT (TIME).
    - **Выходы:** Q (BOOL), ET (TIME).
    - **Принцип работы:**
        1. При появлении переднего фронта (переход с FALSE на TRUE) на входе IN, выход Q немедленно становится TRUE, и таймер начинает отсчет времени.
        2. Выход Q остается TRUE в течение времени PT, независимо от состояния входа IN.
        3. По истечении времени PT выход Q становится FALSE. Повторный передний фронт на IN во время отсчета не влияет на текущий импульс.
    - **Применение:** Формирование коротких управляющих сигналов, запуск однократных действий.

Счетчики используются для подсчета событий (импульсов).

- **CTU (Counter Up) - Нарастающий (суммирующий) счетчик**
    - **Назначение:** Считает количество передних фронтов на входе CU.
    - **Входы:**
        1. CU (BOOL): Вход счета.
        2. R (BOOL): Вход сброса.
        3. PV (INT): Уставка счета (Preset Value).
    - **Выходы:**

1. Q (BOOL): Выход.
2. CV (INT): Текущее значение счетчика (Current Value).

- **Принцип работы:**
  1. Каждый передний фронт на входе CU увеличивает CV на 1.
  2. Когда CV >= PV, выход Q становится TRUE.
  3. Если на вход R подать сигнал TRUE, счетчик сбрасывается (CV становится 0).

- **CTD (Counter Down) - Убывающий (вычитающий) счетчик**
  - **Назначение:** Уменьшает значение счетчика при каждом переднем фронте на входе CD.
  - **Входы:**
    1. CD (BOOL): Вход счета.
    2. LD (BOOL): Вход загрузки уставки (Load).
    3. PV (INT): Уставка.
  - **Выходы:**
    1. Q (BOOL): Выход.
    2. CV (INT): Текущее значение.
  - **Принцип работы:**
    1. При подаче сигнала TRUE на вход LD, текущее значение CV становится равным уставке PV.
    2. Каждый передний фронт на входе CD уменьшает CV на 1 (если CV > 0).
    3. Когда CV <= 0, выход Q становится TRUE.
- **CTUD (Counter Up/Down) - Реверсивный счетчик**
  - **Назначение:** Объединяет функционал CTU и CTD.
  - **Входы:** CU (BOOL), CD (BOOL), R (BOOL), LD (BOOL), PV (INT).
  - **Выходы:** QU (BOOL) – выход суммирующего счета; QD (BOOL) – выход вычитающего счета; CV (INT).
  - **Принцип работы:** CU увеличивает счет, CD уменьшает. QU становится TRUE, когда CV >= PV. QD становится TRUE, когда CV <= 0. R сбрасывает CV на 0, LD загружает PV в CV.
- **SR (Set-Reset) - SR-триггер с приоритетом сброса**
  - **Входы:** S1 (BOOL) - Установка; R (BOOL) - Сброс.
  - **Выход:** Q1 (BOOL).

    - **Принцип работы:**
        - Если S1 = TRUE, то Q1 = TRUE (установка).
        - Если R = TRUE, то Q1 = FALSE (сброс).
        - Если S1 = TRUE и R = TRUE одновременно, приоритет у сброса, Q1 = FALSE.
        - Если S1 = FALSE и R = FALSE, Q1 сохраняет предыдущее состояние.
- **RS (Reset-Set) - RS-триггер с приоритетом установки**
    - **Входы:** S (BOOL), R1 (BOOL).
    - **Выход:** Q1 (BOOL).
    - **Принцип работы:** Аналогичен SR-триггеру, но при одновременной подаче S = TRUE и R1 = TRUE, приоритет имеет установка, и Q1 = TRUE.
- **R_TRIG (Rising Edge Trigger) - Детектор переднего фронта**
    - **Вход:** CLK (BOOL).
    - **Выход:** Q (BOOL).
    - **Принцип работы:** Выход Q генерирует короткий импульс длительностью в один скан, когда вход CLK изменяется с FALSE на TRUE.
- **F_TRIG (Falling Edge Trigger) - Детектор заднего фронта**
    - **Вход:** CLK (BOOL).
    - **Выход:** Q (BOOL).
    - **Принцип работы:** Выход Q генерирует короткий импульс длительностью в один скан, когда вход CLK изменяется с TRUE на FALSE.

Приложение Б. Устранение типовых ошибок компиляции

В данном приложении перечислены наиболее распространенные ошибки, возникающие при компиляции проекта, и способы их устранения. Двойной щелчок по сообщению об ошибке в окне сообщений автоматически переместит вас к месту ее возникновения в коде или схеме.

Код ошибки	**Краткое описание**	**Подробное описание и способ устранения**
C0001	Значение константы выходит за пределы	**Причина:** Вы присвоили переменной или ввели константу, значение которой превышает допустимый диапазон для ее типа данных. Например, для переменной типа `INT` (диапазон от -32768 до 32767) было задано значение `40000`. **Решение:** Исправьте значение на корректное или используйте тип данных с большим диапазоном, например, `DINT`.
C0004	Не является компонентом структуры	**Причина:** Произошла попытка обратиться к полю структуры, которое в ней не определено. Например, `MyStruct.NonExistentField`. **Решение:** Проверьте правильность написания имени поля и убедитесь, что оно объявлено в определении соответствующей структуры.

C0019	Невозможно преобразовать тип	**Причина:** Вы пытаетесь присвоить значение одного типа данных переменной другого, несовместимого типа без явного преобразования. Например, присвоить значение типа REAL переменной типа BOOL. **Решение:** Используйте блоки преобразования типов (например, REAL_TO_INT, INT_TO_BOOL) или проверьте логику присваивания.
C0027	Неопределенный идентификатор	**Причина:** В коде или на схеме используется переменная, которая не была объявлена ни в локальной, ни в глобальной области видимости. **Решение:** Объявите переменную в таблице переменных. Внимательно проверьте правильность написания имени переменной (опечатки).
C0029, C0030	Неверный индекс массива	**Причина:** Программа пытается получить доступ к элементу массива по индексу, который выходит за пределы его объявленного диапазона. Например, обращение к MyArray[10], если массив объявлен как ARRAY [0..9]. **Решение:** Проверьте логику работы с индексами. Убедитесь, что индекс не может стать отрицательным (если не разрешено) или превысить максимальный размер.

C0036	Неверная инициализация массива/структуры	**Причина:** Ошибка синтаксиса при задании начальных значений для массива или структуры. **Решение:** Проверьте правильность формата. Для массива: `MyArray : ARRAY [1..2] OF INT := [10, 20];`. Для структуры: `MyStruct : MyUDT := (Field1 := 5, Field2 := TRUE);`.
C0052	Дублирование определения переменной	**Причина:** Две или более переменные в одной области видимости (например, в одном POU или в глобальных переменных) имеют одинаковое имя. **Решение:** Переименуйте одну из переменных.
C0086	Конфликт адресов	**Причина:** Две или более переменные привязаны к одному и тому же физическому адресу ввода-вывода (например, `%IX0.1.1.0`). **Решение:** Откройте редактор аппаратной конфигурации или таблицы переменных и назначьте конфликтующим переменным уникальные адреса.
[Ошибка]	Отсутствует вход/выход у блока	**Причина:** У функционального блока на схеме FBD или LD не подключен обязательный вход или не используется его выход. **Решение:** Подключите к входу соответствующую переменную или константу. Убедитесь, что результат работы

		блока (выход) используется в дальнейшей логике.
[Предупреждение]	Переменная объявлена, но не используется	**Причина:** Вы объявили переменную, но ни разу не использовали ее в программе. Это не ошибка, но может указывать на забытый или лишний код. **Решение:** Удалите неиспользуемую переменную или используйте ее в логике, если она была объявлена для какой-либо цели.